VIS Übungblatt 2

Fehlertoleranz

Aufgabe 3 (Pflicht, 2 P.)

Ein großer Webmail-Anbieter möchte damit werben, 365 Tage im Jahr verfügbar zu sein, d.h. die Ausfallzeit seines Dienstes muss pro Jahr einen Tag (24 Stunden) unterschreiten. Er verwendet zur Steigerung der Performance und Verfügbarkeit eine Serverfarm aus 150 gleichartigen Webservern, jeweils mit einer Verfügbarkeit von 95 Prozent. Im Hintergrund arbeitet eine gedoppelte Datenbank, die zudem diversitär ausgelegt ist. Der eine Datenbankserver hat 95 Prozent Verfügbarkeit, der andere erreicht 90 Prozent. Die restliche Systemumgebung hat eine ideale Verfügbarkeit (100 Prozent). Bitte beurteilen Sie, ob das Werbeversprechen (höchstens 1 Tag Ausfall pro Jahr) erfüllt werden kann.

Seien A_i die Verfügbarkeit einzelner Komponenten und A_{sys} die Systemverfügbarkeit. Für eine Serienschaltung mit n Komponenten gilt:

$$A_{serie} = \prod_{i=1}^{n} A_i$$

Für eine Parallelschaltung mit n Komponenten gilt:

$$A_{parallel} = 1 - \prod_{i=1}^{n} (1 - A_i)$$

Damit ergibt sich für das beschriebene System eine Gesamtverfügbarkeit von:

$$A_{sys} = (1 - (1 - 0.95)^{150}) \cdot (1 - (1 - 0.9) \cdot (1 - 0.95)) \approx 0.995$$

Also beträgt die durchschnittliche Ausfallzeit des Gesamtsystems

$$(1 - A_{sys}) \cdot 365 \text{ Tage } = 1.825 \text{ Tage}$$

und das Werbeversprechen wird - statistisch berechnet - nicht erfüllt.

Aufgabe 5 (Pflicht, 4 P.)

Ein Dateiserver ist fünf Jahre in Betrieb gewesen und während dieser Zeit insgesamt nur 4 Stunden und 20 Minuten unverfügbar gewesen.

a) *Wie hoch war seine Gesamtverfügbarkeit in Prozent? (Erforderliche Genauigkeit: 3 Nachkommastellen)*
Für die Gesamtverfügbarkeit gilt:

$$A_{gesamt} = \frac{t_{gesamt} - t_{ausfall}}{t_{gesamt}}$$

In Fünf Jahren können entweder 1 Schaltjahr oder 2 Schaltjahre auftreten, d.h. wir unterscheiden:
1 Schaltjahr $\rightarrow 365\text{d} \cdot 5 + 1\text{d} = 1826\text{d}$

$$A_{gesamt_{1ly}} = \frac{1826\text{d} \ - 4,\overline{3}\text{h}}{1826\text{d}} \approx 0.999901 = 99,9901\%$$

2 Schaltjahre $\rightarrow 365\text{d} \cdot 5 + 2\text{d} = 1827\text{d}$

$$A_{gesamt_{2ly}} = \frac{1827\text{d} \ - 4,\overline{3}\text{h}}{1827\text{d}} \approx 0.999901 = 99,9901\%$$

b) *Der Dateiserver soll nun durch einen neuen mit vergleichbarer Verfügbarkeit und Betriebsdauer ersetzt werden. Zur Verbesserung der Performance erwägen Sie den Einsatz eines RAID-0- Systems (Striping, keine Redundanz) mit 4 Festplatten. Die MTBF der eingesetzten Festplatten betrage ca. 50.000 Stunden je Platte, die MTTR betrage 1 Stunde je Platte (inkl. Recovery). Die Verfügbarkeit der restlichen Komponenten sei ideal 100 Prozent. Ist die geforderte Verfügbarkeit mit RAID-0 noch erreichbar? Begründen Sie Ihre Antwort kurz.*

Im RAID-0-System ist der Dateiserver nur Verfügbar, wenn alle n Komponenten verfügbar sind. D.h. für die Verfügbarkeit ergibt sich:

$$A_{gesamt_{raid0}} = \left(\frac{MTBF}{MTBF + MTTR}\right)^n = \left(\frac{50000\text{h}}{50000\text{h} + 1\text{h}}\right)^4 \approx 0.99992 = 99,992\%$$

Kryptographie

Aufgabe 7 (Pflicht 6 P.): Informationstheoretisch sichere Verschlüsselung

a) Wenn der Schlüssel vorher noch nicht verwendet und echt zufällig erzeugt wurde, erfährt der Angreifer nichts über den Klartext. Aus dem Schlüsseltext kann nämlich durch die Wahl des richtigen Schlüssels jeder beliebige Klartext erzeugt werden, da der Schüssel aber zufällig erzeugt ist, erhält der Angreifer auch keine Kenntnisse über diesen.

b) Werden zwei Klartexte m_1, m_2 mit dem gleichen Schlüssel verschlüsselt, können die Schüsseltexte s_1, s_2 genutzt werden um Informationen über die Klartexte zu gewinnen:

$$m_1 \oplus k = s_1$$

$$m_2 \oplus k = s_2$$

$$s_1 \oplus s_2 = (m_1 \oplus k) \oplus (m_2 \oplus k)$$

$$s_1 \oplus s_2 = m_1 \oplus m_2$$

Man erhält also die XOR-Verknüpfung der beiden Klartext $m_x = m_1 \oplus m_2$. Man kann nun Teile von m_x mit sinnvollem Klartext XOR verknüpfen, z.B. häufig vorkommende Wörter oder häufig auftretende n-gramme, bis ein sinnvoller Teilklartext ensteht. Es wird also ein Teil von m_1 bzw m_2 geraten.[1]

c) Wir haben den eben beschriebenen Angriff durchgeführt. Dafür werden alle gegeben Schlüsseltexte zuerst paarweise und das Ergebnis dann mit unserem geratenen *Crib* XOR-verknüpft. Als Crib haben wir z.B. die Wörter *Haus* oder *Kind* verwendet, beim Wort *Baum* erhalten wir folgende Ausgabe:

```
1  ------------XOR Kombinationen von jeweils zwei Schlüsseltexten und Crib--------
2  [0,1] AUMN
3  [0,2] AQSE
4  [0,3] AUTO
5  [0,4] Q[RG
6  [0,5] WQRV
7  [1,2] BEKF
8  [1,3] BALL
9  [1,4] ROJD
10 [1,5] TEJU
11 [2,3] BERG
12 [2,4] RKTO
13 [2,5] TAT^
```

[1] Siehe auch `http://travisdazell.blogspot.de/2012/11/many-time-pad-attack-crib-drag.html`

```
[3,4] ROSE
[3,5] TEST
[4,5] DKU\
```

Man erkennt sinnvolle Klartexte für jede XOR-Kombination, an der das Schlüsselwort 3 (also das 4. Wort aus der Liste) beteiligt ist. Daraus lässt sich schließen, dass der entsprechende Klartext für das Schlüsselwort 3 *Baum* ist. Ein XOR aus Schlüsselwort 3 und dem Klartext *Baum* ergibt also den Schlüssel:

```
------------Schlüssel--------
HUPE
```

Der entsprechende Java Code:

```
public class main {

    final static int[] word0 = {9,0,4,10};
    final static int[] word1 = {10,20,28,9};
    final static int[] word2 = {10, 16,2,2};
    final static int[] word3 = {10,20,5,8};
    final static int[] word4 = {26,26,3,0};
    final static int[] word5 = {28,16,3,17};

    final static String CRIB_TEST_WORD = "BAUM";

    static List<int[]> words = Arrays.asList(word0, word1, word2, word3, word4, word5);

    public static void main(String [] args) {

        System.out.println("------------XOR Kombinationen von jeweils zwei Schlüsseltexten und
             Crib--------" );

        //crib dragging, wir raten, mögliche Klartext(teile). Siehe Aufgabe oben.
        for(int i = 0; i< words.size(); i++){
            for(int j = i + 1; j < words.size(); j++) {
                System.out.println("[" + i + "," + j + "] " +
                       getStringFromAsciiDec(xorWords(xorWords(words.get(i), words.get(j)),
                            cribTestWord())));
            }
        }

        //Crib entspricht word3, also ist CRIB_TEST_WORD xor word3 der Schlüssel.
        System.out.println("\n------------Klartexte--------" );
        for(int i = 0; i< words.size(); i++){
               System.out.println(getStringFromAsciiDec(xorWords(words.get(i), getKey(word3))));
        }

        System.out.println("\n------------Schlüssel--------" );
        System.out.println(getStringFromAsciiDec(getKey(word3)));
    }

    private static int[] getKey(int[] word) {
       return xorWords(word, cribTestWord());
    }

    private static int[] xorWords(int[] word1, int[] word2) {
        int[] result = new int[4];
        for(int i =0; i < word1.length; i++) {
            result[i] = word1[i]^word2[i];
        }
        return result;
    }

    private static int[] cribTestWord() {
       return getAscciiDecFromString(CRIB_TEST_WORD);
    }
```

```
51
52      private static int[] getAscciiDecFromString(String text) {
53          return text.chars().boxed().mapToInt(i->i).toArray();
54      }
55
56      private static String getStringFromAsciiDec(int[] charList) {
57          return Arrays.stream(charList).mapToObj(c ->
                    String.valueOf((char)c)).collect(Collectors.joining());
58      }
59  }
```